The transport layer (L4) primarily deals with error recovery and flow control, but those are only provided by TCP. Both TCP and UDP provide multiplexing via ports—the ability to segregate received segments by port number and give them to the correct application. A segment can also be called an L4PDU (Layer 4 Protocol Data Unit). The L3 IP header protocol field tells whether TCP, UDP, or some other L4 protocol is in use.

Socket—traffic is uniquely identified as belonging to a given conversation or data stream based on a unique combination of IP address, transport protocol (for example TCP or UDP), and an L4 port number.

Connection-Oriented Protocol—TCP sets up a conversation so that data segments are expected and can be noticed as missing if they don't arrive. See the 3-way handshake, described under TCP.

Connectionless Protocol—UDP just blurts out whatever it has to say, hoping that it arrives and that the pace doesn't overwhelm the recipient's buffers.

WELL KNOWN APPLICATIONS AND PORT NUMBERS

Server processes that have a standardized (by the IANA) will open that port on their computer for listening when they start running. Remember that a socket relies both on the choice of TCP or UDP *and* the port number for uniqueness; there's no relationship between TCP and UDP port numbers—only coincidences. Some well known port numbers include:

TCP Port	UDP Port	Application
20		FTP (File Transfer Protocol) Data
21		FTP Control
22		SSH
23		Telnet
25		SMTP
53	53	DNS—Server tells client what IP address corresponds to a domain name. (TCP and UDP are used for different purposes / situations)
	67	DHCP Server
	68	DHCP Client
	69	TFTP (Trivial File Transfer Protocol)—traditionally used to move operating systems and configurations onto Cisco devices.
80		HTTP (Web)
110		POP v3
	161	SNMP—Used for monitoring and control of network devices.
443		SSL
	514	Syslog

TCP

TCP (Transport Control Protocol) [RFC 793]—provides a number of services to applications that UDP doesn't, including

- Error recovery via retransmission—segments are numbered and those not acknowledged are resent
- Ordered Data Reconstruction—the same segment numbers are used to put the pieces of a data stream back in order
- The receiver can manage the pace of transmission (windowing and acknowledging only when ready)
- Connection Establishment—port numbers and beginning sequence numbers are shared before any data is sent

TCP header—20-bytes. Contains no addresses. Don't memorize field names or locations.

Name	Bytes	Description
Source Port	2	
Destination Port	2	
Sequence #	4	Segment number of the segment being sent
Acknowledgement #	4	Segment number of the next expected segment
Offset + (reserved) + Flag Bits	2	
Window	2	
Checksum	2	
Urgent	2	

Notice that acknowledgements use the number of the *next* expected segment. You should also know that sequence numbers are incremented not by 1, but by the size of the segment in bytes.

TCP Connection Establishment—also called the three-way handshake—uses the SYN (SYNchronize) and ACK (ACKnowledge) flags in the TCP header along with the sequence and acknowledgement number fields to agree on a starting value for segment numbering from *each* side. The starting sequence numbers are essentially random and will almost certainly not match. An example handshake between hosts A and B could look like:

Sender	SYN	ACK	Sequence #	Ack #	Description
A	X		734		A tells B it wants to establish a connection and will begin its segment numbering at 734
B	X	X	18,193	754	B will begin numbering *its* segments at 193, synchronizes its expectations for the other side's numbering to 734, and acknowledges segment 734 by asking for the next segment, which will be numbered 754, since there was no body and a TCP header is 20 bytes long.
A		X	754	18,213	A acknowledges that B will begin numbering segments at 193 by asking for the next segment, number 213.

TCP Connection Tear-Down—The FIN (FINished) bit can be used to end a connection.

Windowing—The recipient can tell the sender how much data to send per acknowledgement by putting a value in the window field of the TCP header.

UDP

UDP (User Datagram Protocol)—Has a smaller header than TCP, making it more efficient for uses that don't require or can't use error recovery. For example, in real-time streaming applications, a retransmitted segment would be too late to be of any use. UDP doesn't use the three-way handshake because it is connectionless, again cutting down on overhead and lag for quick notifications and queries. Additionally, UDP doesn't provide windowing, reordering of received data, and no segmentation of large blocks of data; chunks need to already be less than the maximum size.

All UDP provides is best effort delivery of messages with multiplexing via port numbers in a streamlined package.

The UDP Header—only 8 bytes:

Name	Bytes	Description
Source Port	2	
Destination Port	2	
Length	2	
Checksum	2	

Applications

URI (Uniform Resource Identifier)—Also known as a URL (…Locator). Example dissected below:

Scheme	Authority	Path
http://	www.cisco.com	/index.html
(web protocol)	(server's name)	(web page)

Steps to reach a web page:

- DNS—the server name in the URI would be sent to a DNS server for resolution into an IP address. This is a UDP request on port 53.
- TCP handshake establishes connection between client (browser) and web server
- Browser sends an HTTP "get" request
- Server sends back the file (page)